Claims

We claim:

1. A partitioned database system, comprising:

a plurality of storage facilities, each storage facility including data representing a plurality of table rows;

wherein table rows in each storage facility that correspond to a specific table are logically ordered according to a row identifier (row ID);

the row ID comprises a first value based on one or more columns of the table and a second value based on one or more columns of the table; and

the first value of the row ID is predominate in determining the order of the rows in the storage facilities and the second value determines the order of those rows with identical first values.

2. The partitioned database system of claim 1, wherein:

the row ID further comprises a third value and the third value determines the order of rows with identical first and second values.

3. The partitioned database system of claim 2, wherein the row ID is 64 bits.

4. The partitioned database system of claim 3, wherein the first value is specified by 8 bits, the second value is specified by 32 bits and the third value is specified by 24 bits.

5. The partitioned database system of claim 2, wherein the row ID is 80 bits.

6. The partitioned database system of claim 5 wherein the first value is specified by 16 bits, the second value is specified by32 bits, and the third value is specified by 32 bits.

7. The partitioned database system of claim 2, wherein the third value is a uniqueness number that differentiates rows having equal first and second values.

8. The partitioned database system of claim 1, wherein the first value is specified by 16 bits.

9. The partitioned database system of claim 1, wherein:

the first value of the row ID corresponds to ranges of values in a column.

10. The partitioned database system of claim 9, wherein the ranges of values in a column are ranges of dates.

11. The partitioned database system of claim 1, wherein:

the first value of the row ID corresponds to ranges of values derived from at least one column.

12. The partitioned database system of claim 1, wherein

the second value is a value in a specified column.

13. The partitioned database system of claim 1, wherein

the second value is the result of applying a hash function to a value in at least one specified column.

14. The partitioned database system of claim 1, wherein table rows are distributed among the plurality of storage facilities based on the second value.

15. A method for building a partitioned database system, comprising the steps of:

defining columns in a table;

selecting a first group of one or more columns;

selecting a first function based on values in each column of the first group of columns;

selecting a second group of one or more columns;

selecting a second function based on values in each column of the second group of columns;

creating rows of the table;

storing rows of the table in a storage facility in a logical order corresponding to the result of the first function for each row; and

if more than one row of the table has an identical result of the first function, storing those rows in a logical order corresponding to the result of the second function.

16. The method of claim 15, further comprising the step of:

if more than one row of the table has identical results of the first and second functions, storing those rows in a logical order corresponding to a third value for each of those rows.

17. The method of claim 15, wherein the results of the first and second functions for each row are included in a row ID with the third value for each row.

18. The method of claim 16, wherein the third value is a unique value within the set of rows with the same first and second values.

19. The method of claim 15, wherein the step of storing rows comprises:

distributing rows of the table to a plurality of storage facilities based on the result of the second function.

20. The method of claim 15, wherein the first function assigns a value based on one more columns to each of several ranges.

21. The method of claim 20, wherein the first function assigns a value to each of several ranges of values in a date column.

22. The method of claim 15, wherein the second function is a hash function.

23. The method of claim 15, wherein the results of the first and second functions for each row are included in a row ID.

24. A computer program, stored in a tangible medium, for building a partitioned database system the program comprising executable instructions that cause a computer to:

store column definitions for a table;

store a first function based on values of a first group of one or more columns;

store a second function based on values of a second group of one or more columns;

receive data for rows of the table;

store rows of the table in a storage facility in a logical order corresponding to the result of the first function for each row; and

if more than one row of the table has an identical result of the first function, store those rows in a logical order corresponding to the result of the second function.

25. The computer program of claim 24, where the program further includes executable instructions that cause a computer to:

if more than one row of the table has identical results of the first and second functions, store those rows in a logical order corresponding to a third value.

26. The computer program of claim 25, wherein the third value is a unique value within the set of rows with the same first and second values.

27. The computer program of claim 24, where the computer stores rows by:

distributing rows of the table to a plurality of storage facilities based on the result of the second function.

28. The computer program of claim 24, wherein the first function assigns a value based on one more columns to each of several ranges.

29. The computer program of claim 28, wherein the first function assigns a value to each of several ranges of values in a date column.

30. The computer program of claim 24, wherein the second function is a hash function.

31. A method for storing a row identification (row ID) in a data row structure comprising the steps of:

setting a state of at least one bit in a header of a data row based on whether the row is part of a partitioned or unpartitioned table;

including in the header a first portion of the row ID;

if the state of the at least one bit indicates a partitioned table, including a second portion of the row ID in a body of the data row; and

if the state of the at least one bit indicates a nonpartitioned table, specifying a second portion of the row ID that is assumed for the data row.

32. The method of claim 31, wherein the second portion includes a first value and the first portion includes a second value and a third value.

33. The method of claim 32, wherein the second value is the result of a hash function value based on at least one column of the row.

34. The method of claim 32, wherein the first value is a partition value based on at least one column of the row.

35. The method of claim 32, wherein the third value is a system-generated uniqueness number.

36. The method of claim 31, wherein the at least one bit is a single bit that has an initialization state that is changed to a second state when the data row is part of a partitioned table, but is not changed if the data row is part of a nonpartitioned table.

37. The method of claim 31, further comprising the step of creating a row descriptor that indicates how to calculate the second portion of the row ID.